Swift - Data Types

While doing programming in any programming language, you need to use different types of variables to store information. Variables are nothing but reserved memory locations to store values. This means that when you create a variable, you reserve some space in memory.

You may like to store information of various data types like string, character, wide character, integer, floating point, Boolean, etc. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.

## **Built-in Data Types**

Swift 4 offers the programmer a rich assortment of built-in as well as user-defined data types. The following types of basic data types are most frequently when declaring variables −

* **Int or UInt** − This is used for whole numbers. More specifically, you can use Int32, Int64 to define 32 or 64 bit signed integer, whereas UInt32 or UInt64 to define 32 or 64 bit unsigned integer variables. For example, 42 and -23.
* **Float** − This is used to represent a 32-bit floating-point number and numbers with smaller decimal points. For example, 3.14159, 0.1, and -273.158.
* **Double** − This is used to represent a 64-bit floating-point number and used when floating-point values must be very large. For example, 3.14159, 0.1, and -273.158.
* **Bool** − This represents a Boolean value which is either true or false.
* **String** − This is an ordered collection of characters. For example, "Hello, World!"
* **Character** − This is a single-character string literal. For example, "C"
* **Optional** − This represents a variable that can hold either a value or no value.
* **Tuples** − This is used to group multiple values in single Compound Value.

We have listed here a few important points related to Integer types −

* On a 32-bit platform, Int is the same size as Int32.
* On a 64-bit platform, Int is the same size as Int64.
* On a 32-bit platform, UInt is the same size as UInt32.
* On a 64-bit platform, UInt is the same size as UInt64.
* Int8, Int16, Int32, Int64 can be used to represent 8 Bit, 16 Bit, 32 Bit, and 64 Bit forms of signed integer.
* UInt8, UInt16, UInt32, and UInt64 can be used to represent 8 Bit, 16 Bit, 32 Bit and 64 Bit forms of unsigned integer.

## **Bound Values**

The following table shows the variable type, how much memory it takes to store the value in memory, and what is the maximum and minimum value which can be stored in such type of variables.

|  |  |  |
| --- | --- | --- |
| **Type** | **Typical Bit Width** | **Typical Range** |
| Int8 | 1byte | -127 to 127 |
| UInt8 | 1byte | 0 to 255 |
| Int32 | 4bytes | -2147483648 to 2147483647 |
| UInt32 | 4bytes | 0 to 4294967295 |
| Int64 | 8bytes | -9223372036854775808 to 9223372036854775807 |
| UInt64 | 8bytes | 0 to 18446744073709551615 |
| Float | 4bytes | 1.2E-38 to 3.4E+38 (~6 digits) |
| Double | 8bytes | 2.3E-308 to 1.7E+308 (~15 digits) |

## **Type Aliases**

You can create a new name for an existing type using **typealias**.

typealias Feet = Int

var distance: Feet = 100

print(distance)

## **Type Safety**

Swift 4 is a type-safe language which means if a part of your code expects a String, you can't pass it an Int by mistake.

As Swift 4 is type-safe, it performs type-checks when compiling your code and flags any mismatched types as errors.

var varA = 42

varA = "This is hello"

print(varA)

## **Type Inference**

Type inference enables a compiler to deduce the type of a particular expression automatically when it compiles your code, simply by examining the values you provide. Swift 4 uses type inference to work out the appropriate type as follows.